(b) We can make use of various preprocessor directives such as **#define**, **#include**, **#ifdef - #else - #endif**, **#if** and **#elif** in our program.

(c) The directives like **#undef** and **#pragma** are also useful although they are seldom used.

# Exercise

**[A]** Answer the following:

(a) What is a preprocessor directive

1. a message from compiler to the programmer
2. a message from compiler to the linker
3. a message from programmer to the preprocessor
4. a message from programmer to the microprocessor

(b) Which of the following are correctly formed **#define** statements:

```
#define   INCH PER FEET  12
#define   SQR (X)  ( X * X )
#define   SQR(X)   X * X
#define   SQR(X)  ( X * X )
```

(c) State True or False:

1. A macro must always be written in capital letters. X
2. A macro should always be accomodated in a single line. X
3. After preprocessing when the program is sent for compilation the macros are removed from the expanded source code. ✓
4. Macros with arguments are not allowed. X
5. Nested macros are allowed. ✓
6. In a macro call the control is passed to the macro. X

(d) How many **#include** directives can be there in a given program file? As many as you want

(e) What is the difference between the following two **#include** directives:

a) #include "conio.h"
b) #include <conio.h>

a) The compiler first searches for the header file in the current directory if not found then the standard system directories

b) The compiler only searches in the standard system directories

(f) A header file is:

1. A file that contains standard library functions
2. A file that contains definitions and macros
3. A file that contains user - defined functions
4. A file that is present in current working directory

(g) Which of the following is not a preprocessor directive

1. #if
2. #elseif
3. #undef
4. #pragma

All are preprocessor directive

(h) All macro substitutions in a program are done

1. Before compilation of the program
2. After compilation
3. During execution
4. None of the above

(i) In a program the statement:

```
#include "filename"
```

is replaced by the contents of the file “filename”

1. Before compilation
2. After Compilation
3. During execution
4. None of the above

**[B]** What would be the output of the following program:

(a)
```
main( )
{
    int  i = 2 ;
    #ifdef DEF
        i *= i ;
    #else
        printf ( "\n%d", i ) ;
    #endif
}
```

OUTPUT

2

(b)
```
#define PRODUCT(x) ( x * x )
main( )
{
    int  i = 3, j ;
    j = PRODUCT( i + 1 ) ;
    printf ( "\n%d", j ) ;
}
```

OUTPUT

16

(c)
```
#define PRODUCT(x) ( x * x )
main( )
{
    int  i = 3, j, k ;
    j = PRODUCT( i++ ) ;
    k = PRODUCT ( ++i ) ;

    printf ( "\n%d %d", j, k ) ;
}
```

OUPUT

9  25

(d)
```
# define SEMI ;
main()
{
  int p = 3 SEMI ;
  printf ( "%d", p ) SEMI
}
```

OUTPUT

3

**[C]** Attempt the following:

(a) Write down macro definitions for the following:

1. To test whether a character entered is a small case letter or not.
2. To test whether a character entered is a upper case letter or not.
3. To test whether a character is an alphabet or not. Make use of the macros you defined in (1) and (2) above.
4. To obtain the bigger of two numbers.

(b) Write macro definitions with arguments for calculation of area and perimeter of a triangle, a square and a circle. Store these macro definitions in a file called "areaperi.h". Include this file in your program, and call the macro definitions for calculating area and perimeter for different squares, triangles and circles.

(c) Write down macro definitions for the following:

1. To find arithmetic mean of two numbers.
2. To find absolute value of a number.
3. To convert a uppercase alphabet to lowercase.
4. To obtain the bigger of two numbers.

(d) Write macro definitions with arguments for calculation of Simple Interest and Amount. Store these macro definitions in a file called "interest.h". Include this file in your program, and use the macro definitions for calculating simple interest and amount.